USING OBJECT-ORIENTED ANALYSIS TO DESIGN A MULTI-MISSION GROUND DATA SYSTEM

Peter Shames
Jet Propulsion Laboratory
Multi-mission Image Processing Laboratory
Pasadena, California

Abstract

This paper describes an analytical approach and descriptive methodology that is adapted from Object-Oriented Analysis (OOA) techniques. The technique is described and then used to communicate key issues of system logical architecture. The essence of the approach is to limit the analysis to only service objects, with the idea of providing a direct mapping from the design to a client - server implementation. Key perspectives on the system, such as user interaction, data flow and management, service interfaces, hardware configuration, and system and data integrity are covered. A significant advantage of this service-oriented approach is that it permits mapping all of these different perspectives on the system onto a Single common substrate. This services substrate is readily represented diagramatically, thus making details of the overall design much more accessible.

1.0 Context

The Multi-mission Image Processing System (MIPS) performs science instrument data processing and archival product generation for high data rate instruments flown on a wide variety of planetary missions. The system now supports the Galileo, Cassini, and Mars Pathfinder missions and is planned for support of a number of Discovery-class missions. It is part of the Advanced Multi-Mission Operations System (AMMOS) developed at the Jet Propulsion Lab (JPL).

The design integrates a variety of vendor platforms, commercial sub-systems, and customized components into an evolvable system capable of handling a broad range of spacecraft data rates and processing requirements. Support is provided for both local and remote users via workstations, and for remote users without local processing capabilities. Major parts of the system are portable and re-configurable, and a substantial legacy of image processing, analysis, and display applications, called VICAR, are being ported as part of the development task.

The system has just been redesigned from a centralized, VAX- based architecture to a Unix-based, fully distributed, client server architecture. Part of the intent of the re-design was to move the system into these more modern technologies and to make the new system more easily adapted to new mission processing profiles. Modeling, benchmarking, and testbedding have been extensively used to demonstrate feasibility and the design has successfully passed Critical Review.

2.0. Motivation

At the start of the MIPS re-design project there was no overall description of the existing architecture of the system nor of how existing elements, and the new ones that were contemplated, would integrate into the final design. There was, however, a substantial legacy of existing code and sub-system components to be integrated that had been part of the 2'nd generation VAX system. While most of the technical and operations staff were familiar with VAX systems, only a few understood Unix and distributed client-server systems.

The overall system design had to be accessible from several perspectives, starting with the user/operator view and including logical, physical, and functional ones. Hardware, software, user services, networks, security, and access control elements all needed to be made Visible and presented in context. Existing components and processes had to be shown within the context of the new architecture and interfaces. Accordingly, a major part of the system design effort went into defining the overall architecture, communicating that design to the senior user and technical staff, and then refining the design with their active involvement .

The initial approach used high-level dataflow diagrams and context diagrams to describe the overall system design. This was useful from an analytical perspective, but did not convey the design with enough clarity to those who were unfamiliar with client-server architectures. Following some research into object-oriented analysis (OOA) and design techniques the present approach was developed. This approach was strongly influenced by the techniques defined by Coad and Yourdon in their book Object Oriented Analysis, but uses a service-oriented perspective instead of a pure object-oriented one, and acids analytical framework ancl descriptive and representational approaches that have pragmatic applicability for a broad class of re-design efforts.

This paper describes the adaptation of this Object -Orient ccl Analysis (OOA) technique and then demonstrates using it to communicate key issues of system logical architecture. Kcy perspectives on the system, such as user interaction, data flow and management, clicmt-serve.r interfaces, hard ware configuration, ant] system ant] data integrity arc covered. A significant advantage of this service-oriented approach is that it permits mapping all of these different perspectives on the system onto a single common substrate. This services substrate is readily represented diagrammatically, thus making details of the overall design much more accessible.

3.0 Genesis of the Technique

The 00A technique presented by Coad and Yourdon is applicable to a complete system analysis task where a basic design is being newly developed, but where there may be some information derived from previous designs. They describe an analytical process that involves identification of objects, their structure, the subjects, their attributes, and final ly the services performed by the system. in their approach objects arc understood to encapsulate attributes and services on those attributes; objects with message passing interfaces arc the basic entities that appear in the design abstraction.

Coad and Yourdon describe objects as the least volatile elements of the design and posit them as a stable framework upon which to base the rest of the analysis and specification strategy. Other items in this framework, data, sequencing of functions, the functions themselves, and the interfaces among system components arc seen as having increasingly higher levels of volatility. Object definitions arc proposed as the basis of the analysis, and these definitions include a specification of the services provided 1)y each object.

Interfaces have a high volatility in this view because they are implemented as messages passed between objects. The message passing funct ions are a stable part of the design, but the message formats are bound late in the analysis and remain fluid.

The process of identifying objects involves looking for logical structures, other external systems, devices, events that must be remembered, roles played, locations, and organizational units. These elements arc each reviewed to identify common services, att I ibutes, remembered information, and requirements. Services, in their technique, are specified last because they are seen as being highly volatile, including not just processes but also the details of process sequencing. A service is simply defined as "the processing to be performed upon receipt Of a message".

Services arc identified during system analysis 1)y considering the behavior of objects and the basic functions each provides. Behaviors include:

Immediate causation - state, event, response
History - object life history
Function - fundamental services

'1'1 ıe basic services that an object provides may include:

Occur (instance add, change, delete, or select)

Calculate (create, process, alter, derive)
Monitor (observe, record, track)

Where a new system is being designed from the ground up, or an old system is being completely redesigned, such a top-down object-oriented analysis is likely to be extremely useful, especially if the implementation is to be done in an object-oriented language such as C++ or ADA. It is less clear how a complete object-oriented analysis will benefit a more modest re-implementation or system transition project such as the one that prompted this activity. The resulting OO design may be highly descriptive of the problem, but mapping this design onto existing services and legacy applications is likely to be problematic.

The heart of the modified approach is to adopt a client-server architecture and to fores the analysis on only one class of object, Services. A client-server architecture maps cleanly onto the services perspective, providing a useful analytical and descriptive framework.

3.1 Client-Server Architectures

Client-server implementations are now almost commonplace and many elements in modern networked systems use this technology. Many capabilities provided in distributed computing environments, such as networked services, database servers, file systems, and information resources, all incorporate some form of client-server technology. Common services like the TCP/IP remote login (telnet) and file transfer protocol (FTP), the Sun Microsystems developed Network File System (NFS) service, and commercial relational DBMS servers like those from Sybase and Oracle all use client-server approaches. A major component in the Open Software Foundation (OSF) system is the Distributed Computing Environment (DCE), a framework for developing client-server applications,

~lient-server approaches are "in" at the practical level of developing and integrating systems based upon legacy applications, commercial server offerings, and current technology, but there are no well accepted approaches for defining and describing such systems. The older design technologies, such as those which were useful for more centralized implementations, fail to capture much of the richness of a client-server approach or to provide an integrated description of the system. Key elements here are:

- Distributed nature of the system
- Encapsulation of functions and data
- Stable and clearly specified interfaces between components
- Reuse of services and interfaces in the design

Many of these elements are considered in an object-oriented approach, the key difference here being a focus upon services as a basic substrate, rather than all other classes of objects. From this perspective services are relatively stable entities, tied to fixed interfaces and protocols, existing applications and/or capabilities, and commercial servers. The volatility and adaptability in such a system tends to be embodied in new processing algorithms and in the scripts that drive the system, rather than in the definition of the services themselves.

Particularly in our systems, which are used for science data analysis as well as production processing, we assume that the scripts will change from moment to moment (during interactive analysis) or that they may persist for the duration of some project (production data processing). Most of the services are stable over time, though specific details of database content or even data type will change. Where processing details must also change the services framework provides the means for accommodating new user interfaces, algorithms, and data formats. Much of the processing for new projects or new products can be accommodated by adapting existing components and scripting or adapting the processing scenario.

3.2 Basic Service Element

In this approach the essential building block is a basic service element (BSE). Figure 1 provides a diagrammatic view of this model. Each service element consists of the software, hardware, and staff that are required to carry out that function. Each service element has a defined protocol interface and three primary protocol elements: request, response, and fulfillment. In most cases the protocol has an electronic implementation with appropriate protocol data units, but the model does not preclude such options as a written request and a written response or a verbal request and physical media shipped in fulfillment.

and a file defining the processing to be carried out and the account to be charged.

Response - the reply or answer to the request, stated in an electronic message or other medium of exchange. For a film recorder service the response consists of a message that either accepts the request or denies it for some reason, such as inadequate funds or privileges.

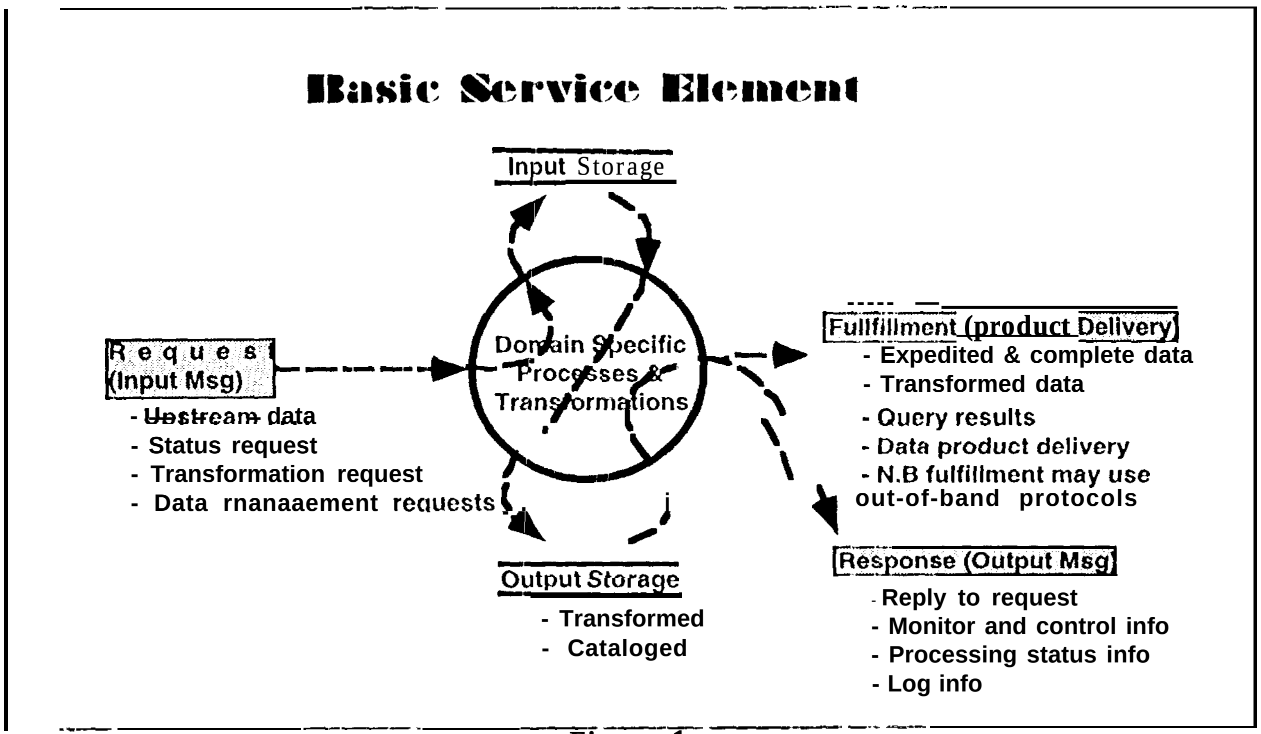


Figure 1

The components of a basic service element are defined as follows:

Protocol - a definition of the process by which requests are made and response and/or fulfillment of the request is provided.

Protocol Data Units - the actual items that are passed during a protocol exchange. May be defined electronic message formats, physical media, or other items of exchange.

Request - the service that is being asked for, stated in an electronic message or other medium of exchange. For a film recorder service the request consists of a data file

Fulfillment - the completion of the request or delivery of a product, often in some other medium of exchange than electronic. For a film recorder service the fulfillment consists of delivery, via mail, of the requested film products.

In many cases the fulfillment of the request may be combined with the response. An example of this might be the return of a the requested rows from a table in response to a database query.

Each service element provides request-driven delivery of data, supports multiple concurrent clients, and uses simple data driven queries. Each element provides its own transform functions, local buffer or cache, and implementation approach, based upon requirements. The Internal implementation choices are opaque to the external user and two implementations of the same service, meeting the same interface protocols, may be entirely different internally.

3.3 Analyzing System Services

The analysis process begins by identifying the basic processes and transformations that the system embodies. This includes an analysis of the essential structure of the processes and the data, identifying major data flows, key transformations, and essential products. As shown earlier, the basic services that a system provides will typically include instances of Occur, Calculate, or Monitor functions.

The analysis should identify both services that are essential fOr performing the work of t he syst em (creat ing products of whatever sort) and services that arc essential for supporting the system (monitoring, tracking. and sustaining. It may prove useful as part of the analysis to group like services, based upon related processes, the class of data handled (image, textual, tabular), the type of service rendered (transformation, storage, monitor), or other characteristics such as temporal relationships or spatial location.

3.4 Developing t he Services Substrate

The analysis of the system into services with well-specificct interfaces is the real meat of this technique, but being able to communicate the result ing design, and its several perspective views, is an essential part of the strategy. The intent here is to make what is a mult i-cl imensional system structure accessible to others, some amount of experimentation with visual representations is likely to be necessary before an arrangement is arrived at that adequately communicates the design.

The process is to create a set of named icons representing the services, or service groups, and then to arrange them in some visually meaningful way. One approach is to organize them by major data flows through the system, so that some sense of process flow is inherent in t he arrangement. Another possible approach is to organize groups of services according to classes of processing (interactive, batch, data driven) or even according to organizational element, if that happens to makes sense in the application cont ext. There has not yet been any useful heuristic identified for facilitating this part of the process.

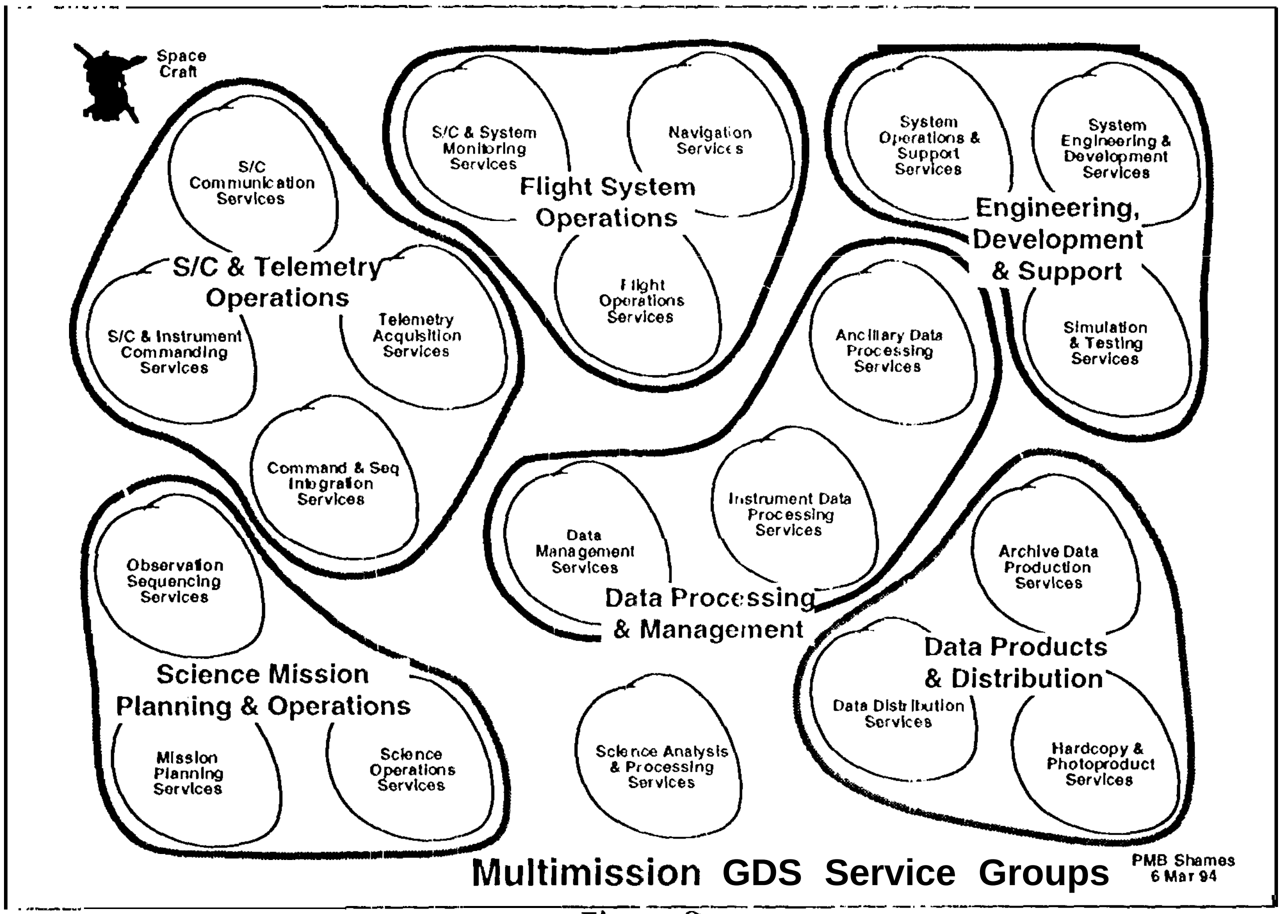
The end result of this effort should be a visual representation of the essential struct urc Of the system, displayed as a SCt Of services or service groups. Each of the entities on this diagram represents one or more instances of a basic service element, with specific processing, local data storage, and associated interface protocols, and protocol data units. This services perspective should show all of the major components of the system, labeled as to the function each provides. This basic services perspective may then be treated as a substrate on which to map many different persp ectives on the system.

Figure 2 shows the result of applying this technique to a high-level analysis of the AMMOS spacecraft ground data system. Clearly each of the "blobs" in this diagram represent more than just one service. In fact they represent groups of associated services, where each group has a related set of fl mct ions and operates on a common set of cl ata. Such a high level diagram is useful for providing contex t for the whole system; lower level decomposition into individual services and interface permits the necessary levels of detail and specificity to be exposed.

3.5 Mapping Other Perspectives

Given the basic services substrate a number of other perspect ives on the system can be presented. One key perspect ive on any major data handling system arc its data flows. One or more overlays, showing major data flows during different processing stages, at different time intervals, or in different operat ing modes can be used to explain the operational characteristics of the system, 1)ifferent processing stages (telemetry handling, assembling level O inst rument frai nes, production of CD-ROMs with calibrat cd data), different processing modes (batch, interact ive, data -driven), or different user perspectives (remote science, production processing, operations) may require quit e different represent at ions or perspect ives.

choices. While this approach provides a useful means of tracking and visualizing alternatives it does not obviate the need for careful system engineering studies, hardware sizing, or configuration analysis.



Space Craft
S/C Communication Services
S/C & Telemetry Operations
S/C & Instrument Commanding Services
Telemetry Acquisition Services
Command & Seq Integration Services
S/C & System Monitoring Services
Navigation Services
Flight System Operations
Flight Operations Services
System Operations & Support Services
System Engineering & Development Services
Engineering, Development & Support
Simulation & Testing Services
Ancillary Data Processing Services
Instrument Data Processing Services
Data Management Services
Data Processing & Management
Archive Data Production Services
Data Products & Distribution
Data Distribution Services
Hardcopy & Photoproduct Services
Observation Sequencing Services
Science Mission Planning & Operations
Mission Planning Services
Science Operations Services
Science Analysis & Processing Services
Multimission GDS Service Groups
PMB Shames 6 Mar 94

Figure 2

The software components that support each service can also be directly mapped onto the basic services substrate, thus showing this perspect ive of the system. Because user interfaces, service APIs, and other componenets may be used in several parts of the total system, re-use of S/W components should be readily evident in this view.

Mapping the system services onto hardware plat forms may also be readily demonst rat cd, thus presenting the other major development aspect of an overall design. It is convenient to experiment with mapping the services design onto different hardware platforms, permitting an easy graphical assessment of different design

The basic substrate may also be used to analyze or represent other elements of an overall process design, such as organizational structures . It has proven inst ructive to map the different organizational entities involved in development, integration, and operation onto t he basic service diagram. This can help clarify a variety of issues with regard to organizational boundaries and interfaces, clearly a part of the analysis of system and processes.

We have experimented with using this basic approach to elucidate other aspects of the overall system, with mixed results. For example, showing the data flows between each service and a centralized monitoring subsystem produces a diagram that looks" like the proverbial plate of spaghettii.

Physical interconnections, such as local or wide-area networks, can also be overlaid on the basic substrate. In practice this has not proven to be too practical, largely for topological reasons related to security and server/network isolation considerations. Attempting to display control domains, particularly where there are complex overlapping access policies. with different styles and modes of access, has also not been too successful.

3.6 Specifying Interfaces and Protocols

The approach just described produces an accessible pictorial representation of the system design, one that is quite readily understood by a broad range of individuals. To complete the system description at a structural level it is also necessary to specify the interfaces, protocols. and protocol data units for each of these service entities. We have used two related organizing approaches for this, one is a simple template, adapted from the, ISO protocol stack, that describes the interface layers for each service, the other is a summary table of services, protocols, and protocol data units. An instance of this template, or mini-System Interface Specification (SIS), is created for each service defined in the design.

The mini-SIS departs from the 7-layer ISO stack in that it differentiates the application layer from an API or library call layer, and it collapses the transport and network layers. The modified stack has the following layers:

Application/Utility
Application Programs: These are programs that are developed as part of the interface, using lower level services and interfaces. Utilities may be provided as part of the interface, either as examples or templates, monitor and edit tools, or as primitive applications that exercise the API.

API/Library
Application Programming Interface (API)/Library: Developed in house or provided by some other means. There may be one or more of these interfaces that are used at different levels in applications (OS interfaces, vendor interfaces, encapsulation interfaces).

Presentation
Data Representation: fundamental data format and representation (binary data formats, VAX, IEEE), also may reference data conversion interface layers such as External Data Representation (XDR).

Session
Authentication and Session Control: establishing connections, identifying communicating parties at each end, managing the session. Handling of userids, permissions, and access control. This could involve passwords, other means of electronically identifying individuals, or a handshake with a live user.

Transport/Network
Reliable End-to-End Data Transfer: means of moving data reliably from initial source to final destination. For networks these are the protocol layers such as the Transmission Control Protocol (TCP) and the Internet Protocol (IP) from TCP/IP. Where physical media such as CD-ROMS are being transfered this might be UPS.

Data Link
Point-to-Point Link: protocol and link layer that provides a point to point connection and control. For networks this is the protocol layer such as Ethernet interface and encoding standards. For physical media this is N/A.

Physical
Actual Physical Medium: the medium which is being used to complete the interface. For networks this is the Ethernet coax, twisted pair, or optical fiber. Where actual physical media are being transfered this might be a CD-ROM or magnetic tape.

The other supporting document is a table describing each server (hardware platform), the services allocated to it, the access controls imposed for each service, and any related security concerns. Separate diagrams may also provided showing the physical hardware configuration, both overall and in detail, the network topology, and the physical layout of the! facility. These diagrams are directly related to the hardware overlay perspective on the system and provide additional details on the servers themselves.

Some other perspectives on the system, such as network topology,isolation, access controls and other security concerns, appear to be most readily presented on a physical view of the system rather than a services one. By using common nomenclature these views may be directly related to the coherent view of the system provided by the services analysis, thus tying the entire design description together.

developed during the MIPS re-design. The! to]J-level analysis of tile MIPS system identified three major groups of processing services: telemetry anti data processing. data anti information management, anti data product generation. It aiso identified two groups of suppoi t services: operations anti management, and system support, anti a set of services associated with users, either local or remote.

Figure **3 shows** these service groupings anti the individual services contained in each. Note that system engineering, development, anti related support services arc an essential part of MIPS but arc not shown.

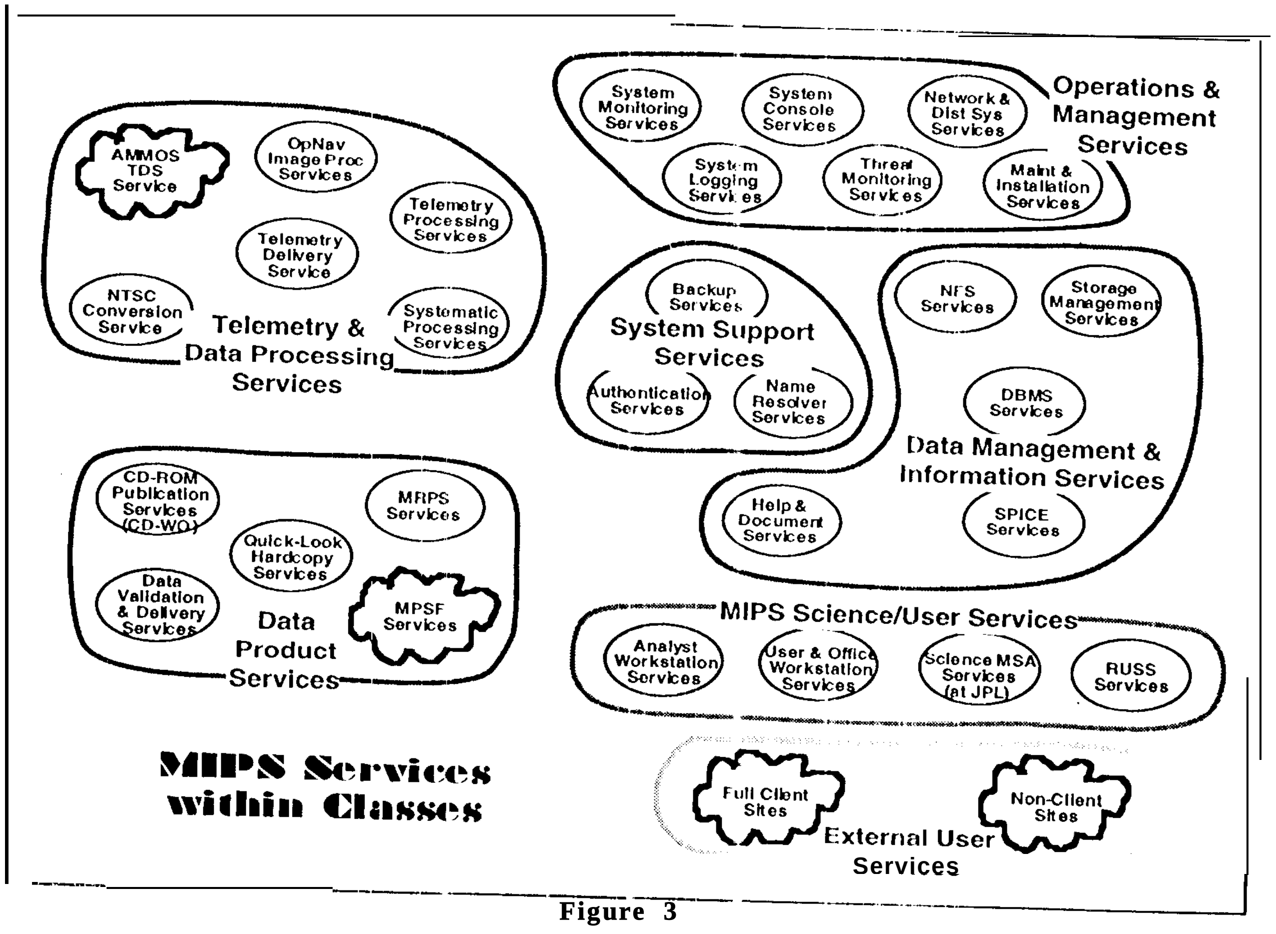

Operations & Management Services
System Monitoring Services
System Console Services
Network & Dist Sys Services
System Logging Services
Threat Monitoring Services
Maint & Installation Services
AMMOS TDS Service
OpNav Image Proc Services
Telemetry Processing Services
Telemetry Delivery Service
NTSC Conversion Service
Systematic Processing Services
Telemetry & Data Processing Services
Backup Services
System Support Services
Authentication Services
Name Resolver Services
NFS Services
Storage Management Services
DBMS Services
Data Management & Information Services
Help & Document Services
SPICE Services
CD-ROM Publication Services (CD-WO)
MRPS Services
Quick-Look Hardcopy Services
Data Validation & Delivery Services
MPSF Services
Data Product Services
MIPS Science/User Services
Analyst Workstation Services
User & Office Workstation Services
Science MSA Services (at JPL)
RUSS Services
MIPS Services within Classes
Full Client Sites
Non-Client Sites
External User Services

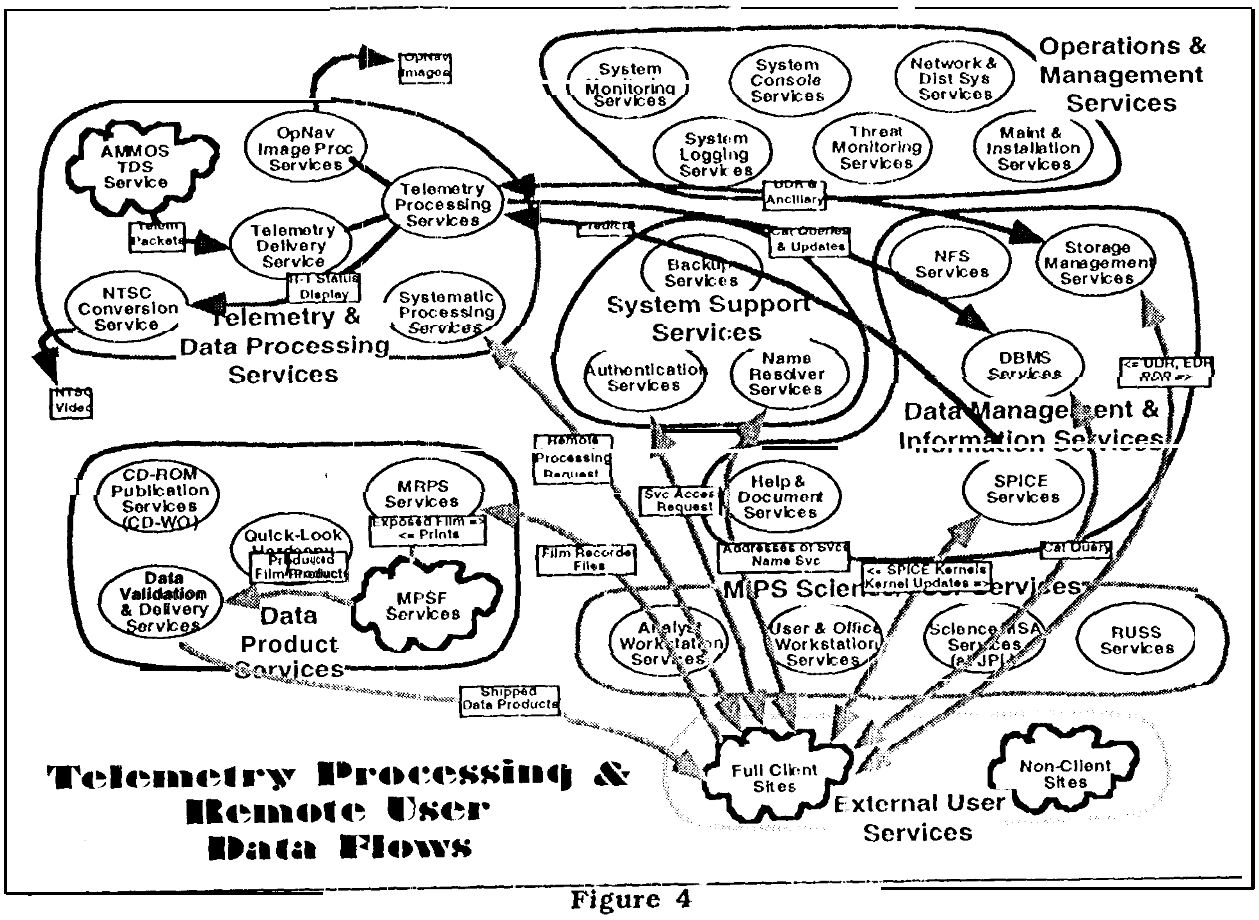
Figure 3

4.0 Applying the Approach to a Real Problem

This section will demonstrate the application of this approach to a real design problem, using some Of the materials

The sei vices are grouped together based upon natural affinities, as should be obvious from the group names. 'l'here are of order 5-7 groups each with roughly 5-7 services defined. This "fingers of onc hand" o r "span of cent ml" rule is commonplace and requires little further discussion. Do note that applying the technique to larger systems is possible, but that higher level aggregations of service groups may be required (as shown in Figure 2).

earlier, the effect of different allocations is easy to visualize using this representation, a capability wc made use of during design trade studies.



Operations & Management Services
System Monitoring Services
System Console Services
Network & Dist Sys Services
System Logging Services
Threat Monitoring Services
Maint & Installation Services
OpNav Images
AMMOS TDS Service
OpNav Image Proc Services
Telemetry Processing Services
Telem Packets
Telemetry Delivery Service
NTSC Conversion Service
RT Status Display
Systematic Processing Services
Telemetry & Data Processing Services
NTSC Video
UDR & Ancillary
Predicts
Cat Queries & Updates
Backup Services
System Support Services
Authentication Services
Name Resolver Services
NFS Services
Storage Management Services
DBMS Services
<= UDR, EDR RDR =>
Data Management & Information Services
Remote Processing Request
Svc Access Request
Help & Document Services
SPICE Services
CD-ROM Publication Services (CD-WO)
MRPS Services
Exposed Film => <= Prints
Quick-Look Hardcopy
Produced Film Products
Film Recorder Files
Addresses of Svcs Name Svc
<= SPICE Kernels Kernel Updates =>
Cat Query
MIPS Science User Services
Data Validation & Delivery Services
MPSF Services
Data Product Services
Analyst Workstation Services
User & Office Workstation Services
Science MSA Services (at JPL)
RUSS Services
Shipped Data Products
Telemetry Processing & Remote User Data Flows
Full Client Sites
Non-Client Sites
External User Services

Figure 4

4.1 Mapping Dataflow, Software, and Hardware

Figure 4 shows two different dataflow overlays on the basic MIPS service substrate. The upper dark arrows in Figure 4 demonst rate telemet ry processing flows during real-tinlc processing. while the lower light arrows show dataflows to a remote user at a workstation. Note that the arcs are labeled as to the key items that flow and that directional arrowheads provide additional information.

Figure 5 shows the mapping of services onto the physical hardware platforms that were used to implement the system. As noted

Figure 6 shows the mapping of services onto the software components that implement the system. Note that there is considerable re- use of software components and this perspective makes these relationships clear.

4.2 Mapping Remaining Perspectives

While it is possible to use this technique to describe other perspectives of the system, not all of them have been as successful as the previous examples. As described in section 3.5, wc have tried to usc this approach to provide representations of monit ori ng subsyst ems, network topology, and access cont rol domains, among others. These diagrams have not proven to be particularly useful when mapped onto the services substrate, so we are not including them.

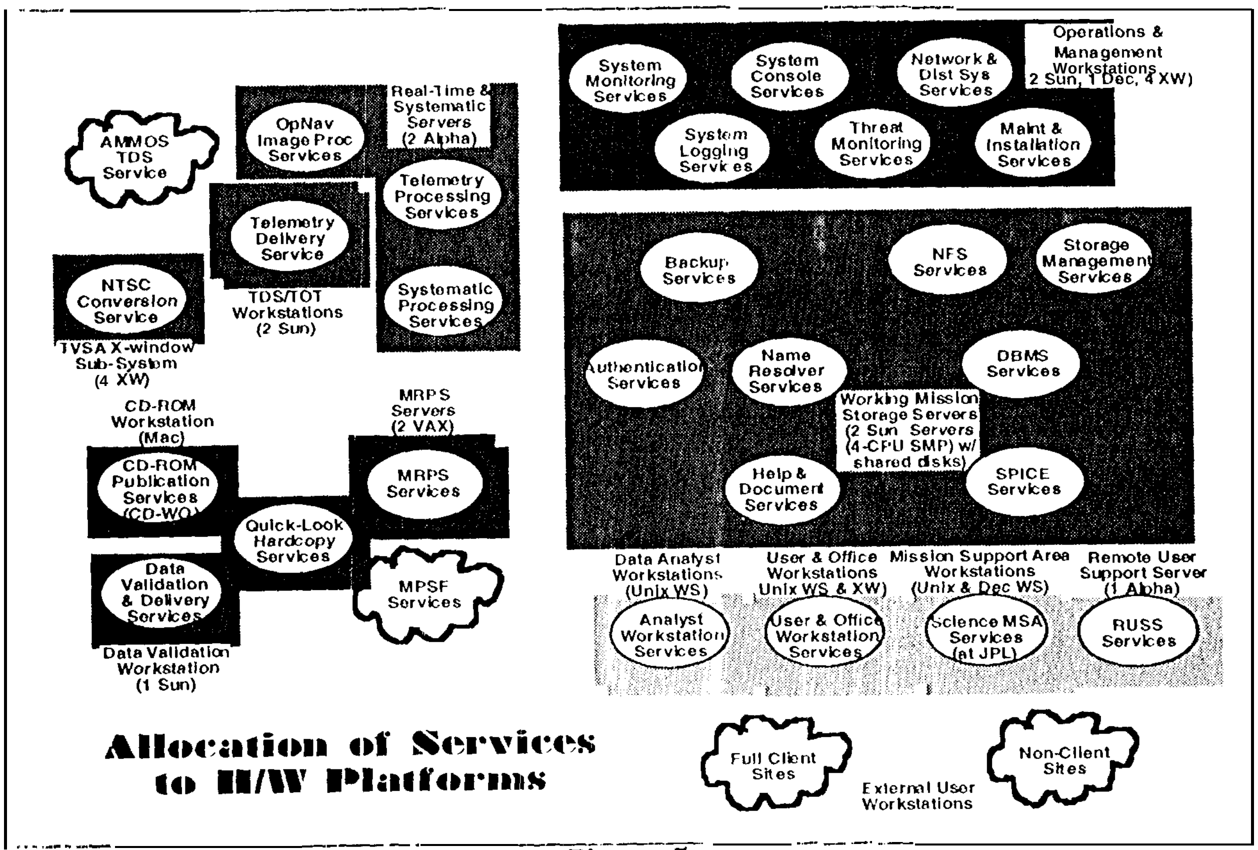

Allocation of Services to H/W Platforms
AMMOS TDS Service
OpNav Image Proc Services
Real-Time & Systematic Servers (2 Alpha)
Telemetry Processing Services
Telemetry Delivery Service
TDS/TOT Workstations (2 Sun)
Systematic Processing Services
NTSC Conversion Service
TVSA X-window Sub-System (4 XW)
CD-ROM Workstation (Mac)
CD-ROM Publication Services (CD-WO)
MRPS Servers (2 VAX)
MRPS Services
Quick-Look Hardcopy Services
Data Validation & Delivery Services
Data Validation Workstation (1 Sun)
MPSF Services
System Monitoring Services
System Console Services
Network & Dist Sys Services
Operations & Management Workstations 2 Sun, 1 Dec, 4 XW)
System Logging Services
Threat Monitoring Services
Maint & Installation Services
Backup Services
NFS Services
Storage Management Services
Authentication Services
Name Resolver Services
DBMS Services
Working Mission Storage Servers (2 Sun Servers (4-CPU SMP) w/ shared disks)
Help & Document Services
SPICE Services
Data Analyst Workstations (Unix WS)
User & Office Workstations Unix WS & XW)
Mission Support Area Workstations (Unix & Dec WS)
Remote User Support Server (1 Alpha)
Analyst Workstation Services
User & Office Workstation Services
Science MSA Services (at JPL)
RUSS Services
Full Client Sites
External User Workstations
Non-Client Sites

Figure 5

For the hardware related perspectives of the design a physical representation, which may be directly related to the hardware overlay, provides a better perspective. Both network topology and physical access control. which are implemented, in large part, at the physical level are best shown on a physical substrate. We have also not produced any representation which is particularly useful for describing one to many relationships, such as those between a monitoring subsystem and each other service.

4.3 Specifying the Service Interfaces

Table 1 shows the complete description of the service interfaces and protocols. Only some of the services are shown in this abbreviated view.

Table 2 shows the access control domain information used to specify in detail which physical servers run which services, who may access these services, and other details about how the security approach is defined and implemented.

A one page mini-SIS that uses the modified ISO template to define all interface layers is developed for each of the service interfaces.

4.4 Adapting the design to new missions

The overall MIPS system has been designed for adaptability and this latest re-design effort has further enhanced that by defining a new client-server architecture, with cleanly established interfaces. Starling from this baseline it has been possible to adapt the system to new missions and to new processing scenarios as required. The architecture accommodates re-use of components and whole sub-systems, depending on the requirements of the new mission.

Most of the work of describing the system adaptation requires changes in only one or two overlays, such as data flows or

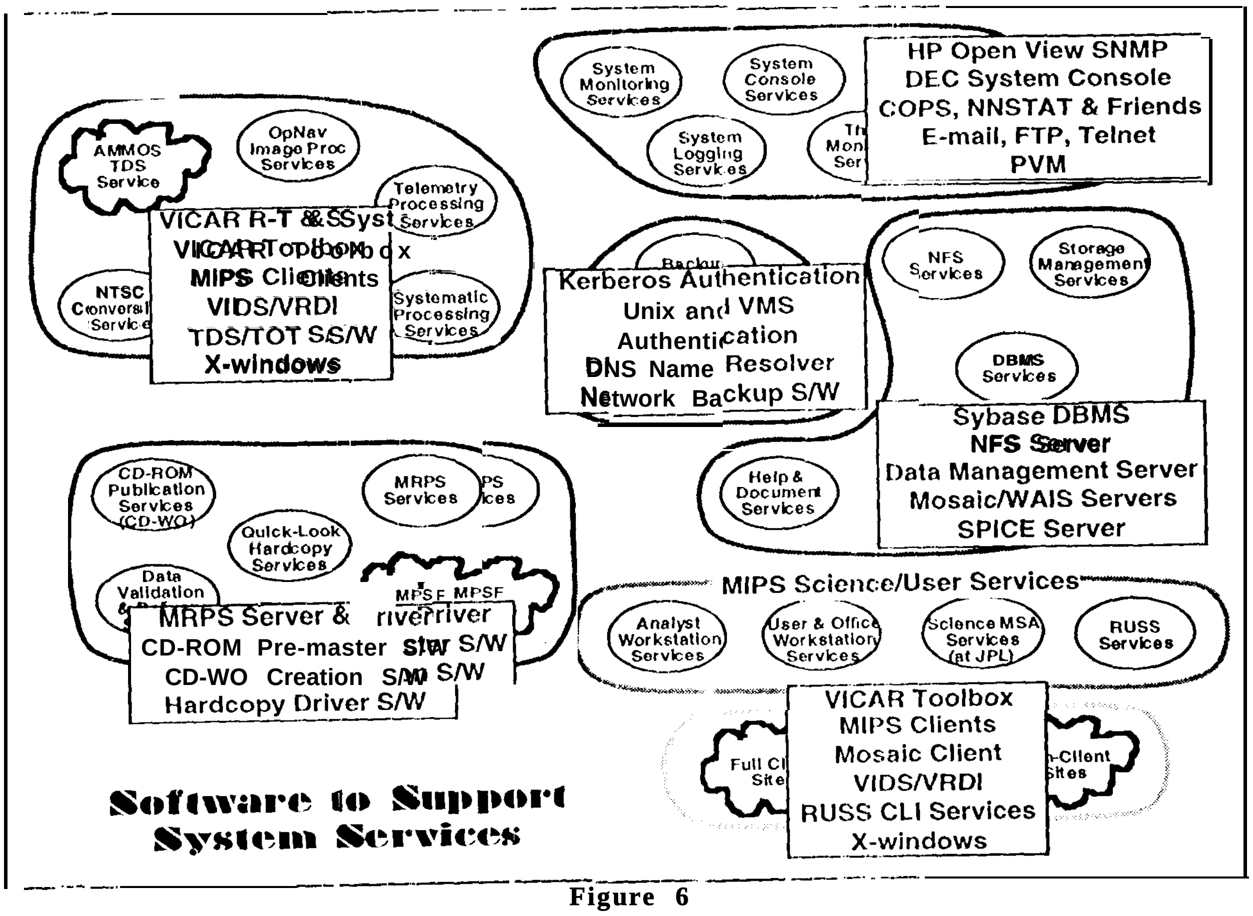


Figure 6

For some new missions, such as Mars Pathfinder with its very low data flows, most of the existing subsystems are used directly. The main telemetry and systematic processing is allocated to its own workstation, but the rest of the services are just adapted to the new data types. For other high rate missions, such as Cassini, some of the hardware platforms will need to be upgraded and larger data caches will be required. None of this affects the basic architecture of the system, though some specific processing algorithms, and many of the scripts, will be re-written to meet the new mission requirements.

Since the service-based design is largely invariant over such re-uses it is easy to use it to manage and describe these adaptations.

hardware platform allocations. The basic system processes, architecture, interfaces and protocols (at several levels), software design, and security structures do not change.

In the actual implementation, or during design trades, it is possible to update an individual service without altering the overall architecture. Because each service has defined interfaces at each of level in the protocol stack (data units, API, representation) it is possible to replace one instance of a service with another. An familiar example would be replacement of one relational DBMS server, which uses SQL and the standard API, with another one that adheres to the same interfaces but has higher performance or improved failover characteristics.

This paper does not deal with many implementation details, but it should be noted that the definition of service interfaces, APIs, and protocols depends heavily upon open systems standards. Our standards include specification of Internet RFCs, POSIX, CCSDS, and Other non-proprietary national and international standards. Over time we expect the client-server design, this design representation, and the use of standards for implementation to pay a high return on investment.

5.0 Conclusion

As has been demonstrated, this design approach works very well for describing the key features of a complicated, distributed system. The technique has been shown to work well for the major MIPS system redesign and it has also been applied to the system analysis and re-design of the much larger AMMOS ground data system. Once the design has been described and the interfaces bound, moving into the implementation phase has been quite straightforward. Because this analytical approach and the associated descriptive technique makes the features of the overall design accessible, it has been valuable in a working group setting as part of the continuing design and implementation process.

The analytical part of the approach provides very complete descriptive facilities for the key perspectives of a multi-faceted system design. The service substrate representation makes much of this information accessible, but has not been able to display all of these perspectives. Some hardware-related views, such as network topology or security, are not easily related to the basic services perspective. We have found, however, that having the basic services analysis and protocol details specified, and the service substrate provided as a representation of other system perspectives, makes the entire design accessible in a way that no other approach seems to provide.

6.0 Acknowledgements

The techniques described in this paper were evolved and tested during the MIPS system re-design. Much support and encouragement were provided by Bill Green and Tom Handley, as well as the other group leaders. Other members of the MIPS Design Team supplied constructive critiques and suggestions. Joe Kahr, John Duxbury, and other members of the Cassini DCS Working Group provided valuable feedback as the technique was applied to the AMMOS analysis and re-design.

Table 1

MIPS Distributed System interlace Specs - PMDS	Interface Protocol Data Units	Interface Protocol
Object == R-T Service		
Request	PDF or Icon Chain requesting processing of predicted S/C data	PDF or Icon Chain presented to Exec
Response	Initiate data ingest process using TDS/TOT commands	TOS Private Protocol
Fullfillment	UDR, Catalog updates, R-T Displays, TVSA images, in appropriate data media or formats	VICAR format files and appropriate service protocols (MDMS, WMS, TVSA)
Object == MRPS Service		
Request	MRPS Request Buffer (File, Processing, Quantity)	MRPS Private (RPC) + VICAR Data File
Response	Accept [ETA]/ Deny	MRPS Private (RPC)
Fullfillment	Photoproduct written to film	Photoproduct is sent to MPSF for processing, QC'ed and Delivered
Object == MDMS Service		
Request	ANSI-SQL formatted DBMS query	SQL Std I/F protocol or MDMS Gateway protocol
Response	SQL formatted reply/ report/ result code/ deny	Std SQL I/F error/ status/ data parameters
Fullfillment		
Object == Mail/SMTP Service		
Request	Process (read/ send/ store/ reply) to RFC822, [MIME] messages	RFC 822, [MIME]
Response	status of process (delivery accomplishment/ failure, [requested reply])	RFC 822 responses
Fullfillment		

Table 2

Server Interface and Access Specification

	ccess Mode					
R-T Server						
Login	Direct Login	Std VMS	Full User facilities	**Ops Users Only ***	Std VMS w/ access control lists	Only Ops users have std VMS accts No access from systems outside of MIPS
FTP	Direct to FTP Daemon	Std VMS	Full FTP capabilities	Ops users & progs		
File System	Direct access only	Std VMS	File file system func	Ops users Only		
Process Invokation	Direct login only	Std VMS	Full system access	Ops users only		
Systematic Server						
Login	Direct Login	Std VMS	Full User facilities	**Ops Users Only**	Std VMS w/ access control lists	Only Ops users have std VMS accts No access from systems outside of MIPS Protocol allows execution of limited set of functions
FTP	Direct to FTP Daemon	Std VMS	Full FTP capabilities	Ops users & progs		
File System	Direct access only	Std VMS	File file system func	Ops users Only		
Process Invokation	Direct login	Std VMS	Full system capabilities	Ops users onlu		
Processing - remote job request	**R-Proc protocol ****	Kerberos	Execute limited set of functions	All W/S & rem users		
MRPS Server(s)						
Login	Direct Login	Std VMS	Full User facilities	**Ops Users Only**	Std VMS w/ access control lists Denial of service if queue fills	Only Ops users have std VMS accts Protocol allows execution of limited set of functions
FTP	Direct to FTP Daemon	Std VMS	Full FTP capabilities	Any local MIPS users		
File System	Direct access only	Std VMS	File file system func	Ops users Only		
Film Recorder Processing	MRPS protocol	**Kerberos *****	Request MRPS services	Any MIPS users		
Database Server						
Login	Direct login/telnet	Std UNIX/NIS	Full user facilities	**Ops Users Only**	Std Unix passwd protection	Only Ops users have std Unix acct. Used by R-T & Syst progs Used by all other progs
FTP	Direct to FTP Daemon	Std Unix/NIS	Full FTP capabilities	Ops users & progs		
File System	Direct access only	Std Unix/NIS	File system func	Ops users Only		
DBlib I/F (Sybase)	Sybase Protocol	Sybase internal	Full Sybase functionality	Ops users only		
VDVS DBVS I/F	VDVS Protocol	Kerberos	Limited DB func (no write/del.?)	Any MIPS users		
Working Mission Storage						
Login	Direct login/telnet	Std Unix/NIS	Full user facilities	**Ops Users Only**	Std Unix passwd protection	Std FTP and NFS services for local users (incl MSA) only Protected access to archive(s) remote users are read-only
FTP	Direct to FTP Daemon	Std Unix/NIS	Full FTP capabilities	Ops users & progs		
File System	NFS	Std Unix/NIS	File system func	Only MIPS internal systems		
WMS Archive I/F	FEI Protocol	Kerberos	RO public archive, R/W private	Any local or remote users		
File Processing	FEI Protocol	Kerberos	File mode conversion/compression	Any local or remote users		
Userstations & Workstations						
Login	Direct login/telnet	Std Unix/NIS	Full user facilities	Any local MIPS users	Std Unix passwd protection	Includes MSAs at JPL MIPS staff on travel may login for mail and other purposes Separate userid/domain from production systems
FTP	Direct to FTP Daemon	Std Unix/NIS	Full FTP capabilities	Any local MIPS users		
File System	Direct access only	Std Unix/NIS	File system func	Any local MIPS users		
Process Invokation	Direct login	Std Unix/NIS	Full system capabilities	Any local MIPS users		
RUSS						
Login	Direct login/telnet	Std Unix	Full user facilities	Any local or remote users	Std Unix passwd protection Corrupt local files and/or system if root Remote pswds are sent in the clear on network links	Only system open for direct remote user login/access Use local passwd file to isolate remote user accts
FTP	Direct to FTP Daemon	Std Unix	Full FTP capabilities	Any local or remote users		
File System	Direct access only	Std Unix	File system func	Any local or remote users		
Process Invokation	Direct login	Std Unix	Full system capabilities	Any local or remote users		

* Operational users may include analysts, Ops Staff, SAs, or developers, as required

** Implementation details for this function are still TBD, may be handled as part of GUI development

*** Implementation of Kerberos authentication not yet planned for MRPS